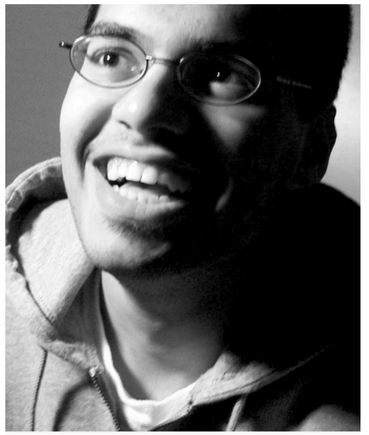
PRACTICAL SOFTWARE-DEFINED UNDERWATER NETWORKS

Prasad, Chinmay, Shiraz
Subnero Pte. Ltd.
Singapore

Global Oceans 2020 - Singapore U.S. Gulf Coast

Prasad Anjangi received his Ph.D. in Electrical & Computer Engineering from National Unversity of Singapore (NUS) in 2016. Prior to that he received the B.Eng. degree in Electronics and Instrumentation Engineering from Andhra University, Andhra Pradesh, India, in 2007 and the M.Eng. degree in Biomedical Engineering from the Indian Institute of Technology (IIT), Bombay, India, in 2009. Currently, he is a Research Scientist at Subnero Pte. Ltd. He worked in semiconductor industries with Atmel and STMicroelectronics as Firmware and Senior Design Engineer, respectively, from 2009 to 2012. His current research interests include underwater acoustic communications, signal processing, networking protocol design, and autonomous underwater vehicles.

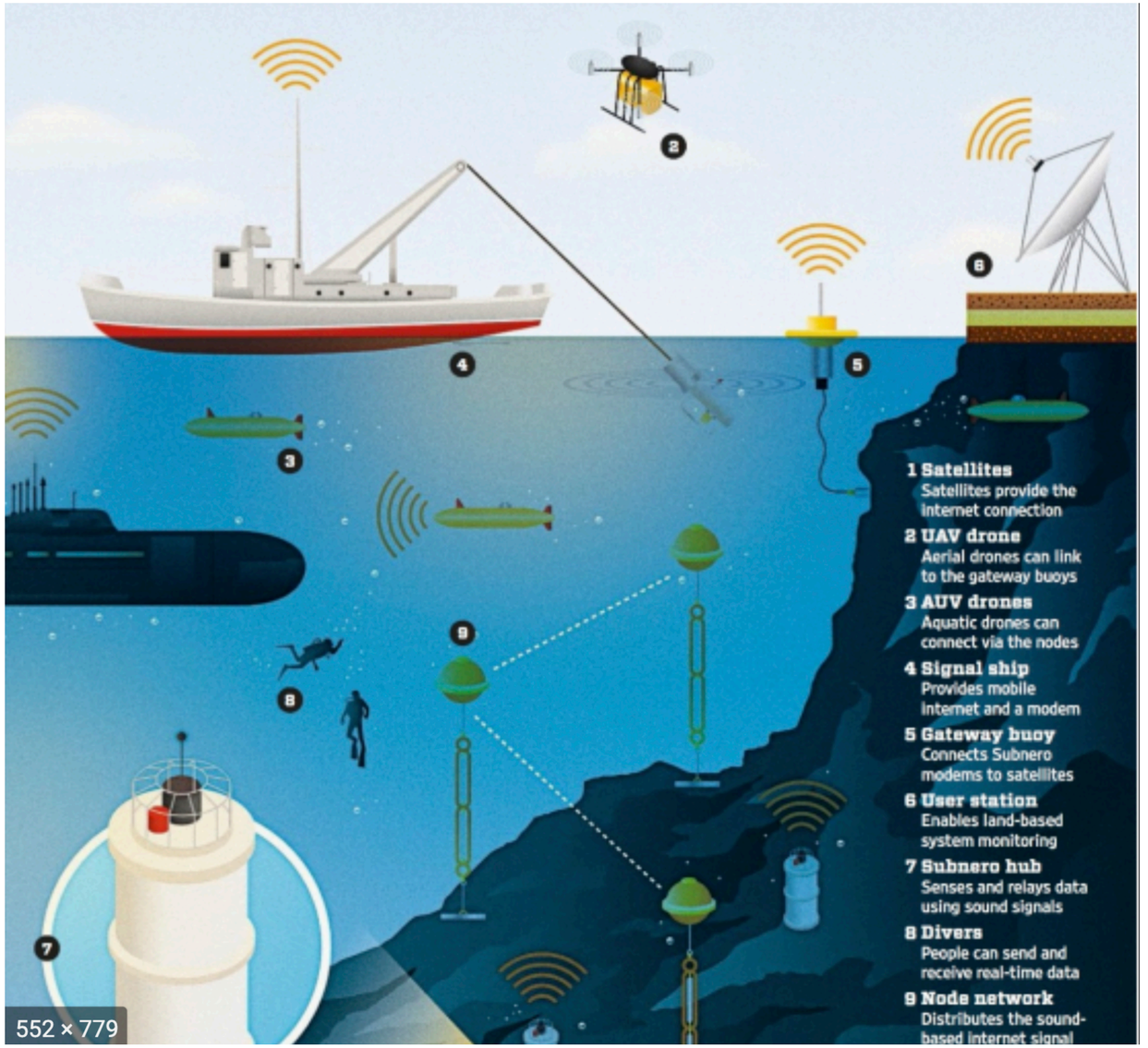
Chinmay Pendharkar received his B.Eng. degree from the National University of Singapore (NUS) in 2006. Since then he has spent more than 10 years in the industry, from working on embedded software in Motorola Electronics Pte. Ltd. to working with experimental audio technologies at a startup spun out of NUS. He also has an M.Sc in Engineering Acoustics from Chalmers University of Technology (Sweden) which he completed in 2011. He is currently the Chief Technology Officer at Subnero, working on constantly improving the technology aspect of Subnero products.



Shiraz Shahabudeen has held various engineering roles including at Infocomm Development Authority of Singapore (IDA), NeST Software, India etc. He was a Research Fellow at ARL, National University of Singapore (NUS) where his research interests included underwater acoustic communications and autonomous underwater vehicles. Currently he works as an independent consultant to NUS, Singapore. Dr. Shahabudeen holds a B.Eng from NUS, M.S in Telecommunication Engineering from Melbourne University (Australia) and a PhD from NUS in Underwater Communications.

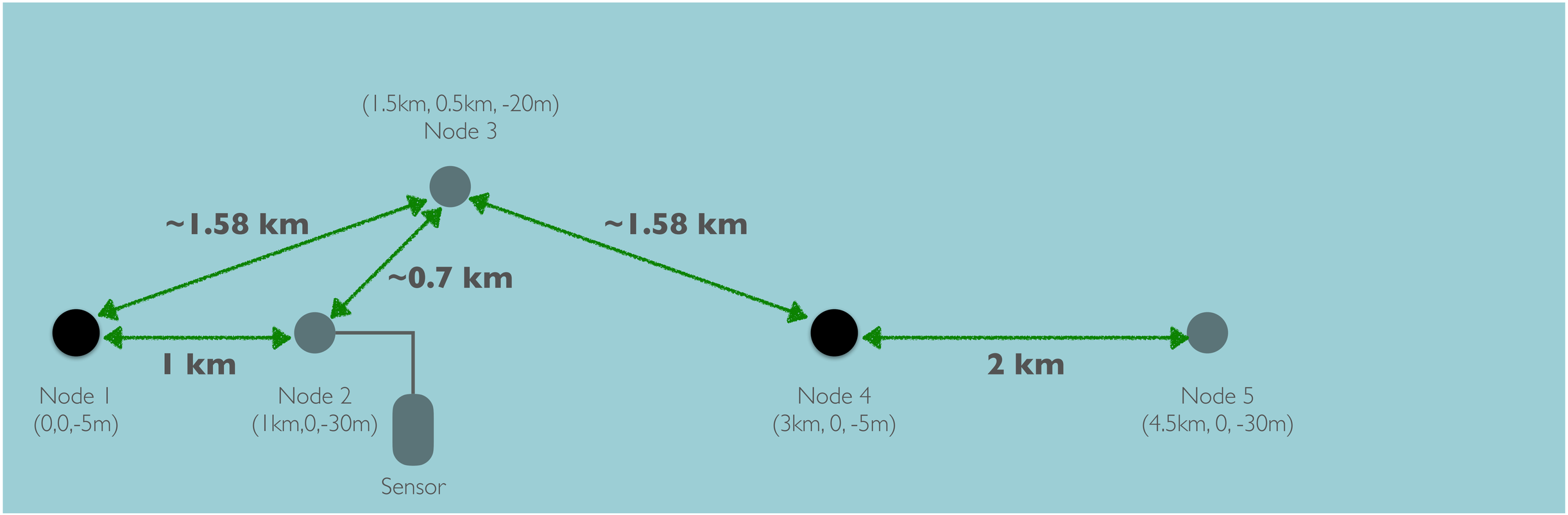
UNDERWATER NETWORKS

1. Satellite (not shown)
2. UAVs
3. AUVs
4. Ships
5. Buoys
6. Land stations
7. Underwater hubs
8. Divers
9. Nodes



- Optical links
- RF links
- GSM links
- Satellite links
- Wired links

AN ILLUSTRATIVE UNET



OUTLINE

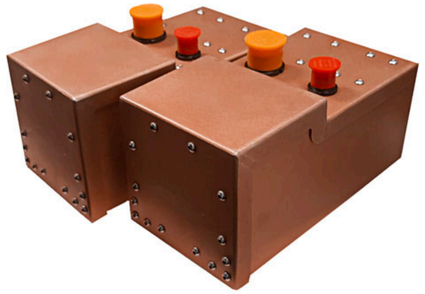
- Introduction - Shiraz
- 2 node point-point networks (PHY, LINK) and 3 node networks (MAC) - Shiraz
- Multihop Routing - Prasad
- Sensors and the Internet - Chinmay
- Localization - Prasad
- Conclusion - mins

INTRO TO UNDERWATER NETWORKS

- [Prof Mandar's video on Unet]

UNDERWATER MODEMS

Embedded

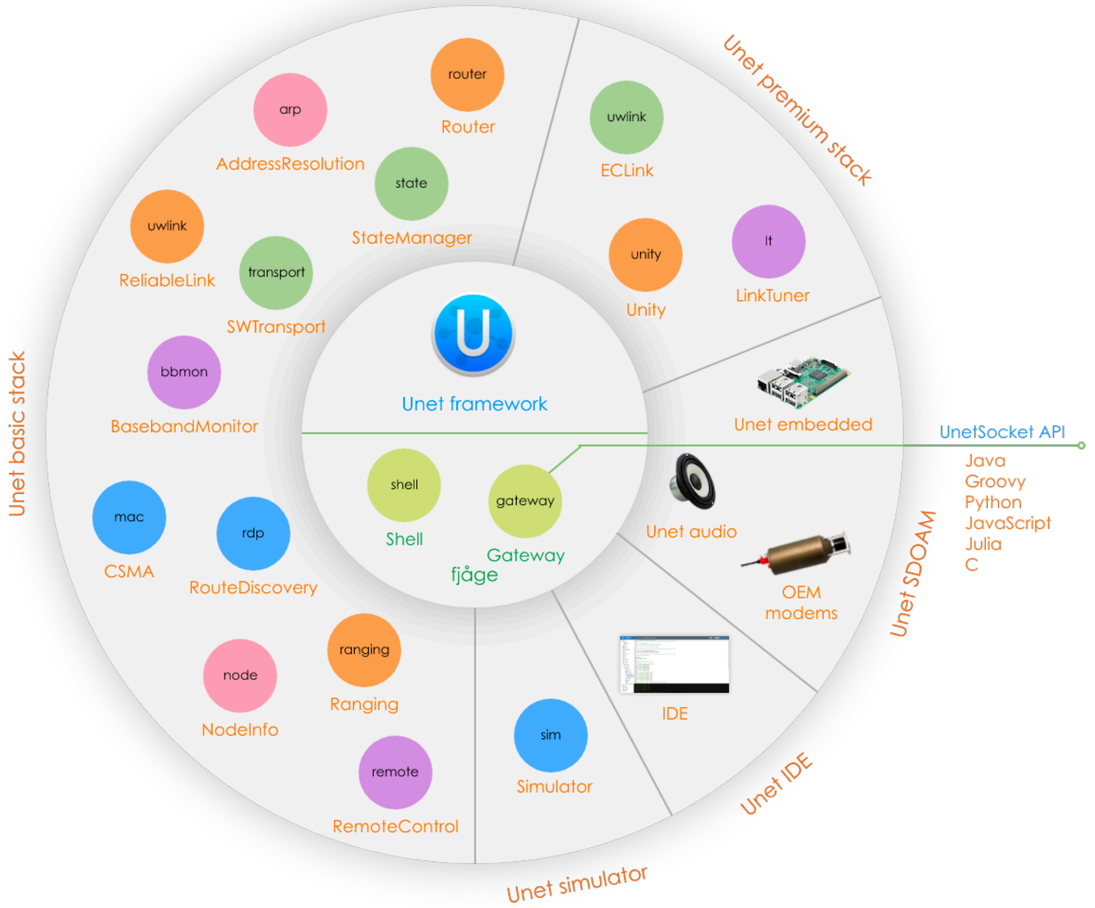


SIMULATORS

- Simulation software
  - SUNSET http://reti.dsi.uniroma1.it/UWSN_Group/index.php?page=sunset
  - DESERT http://desert-underwater.dei.unipd.it/
  - Evologics https://evologics.de/emulator
  - UnetStack https://unetstack.net/
- Using PC audio

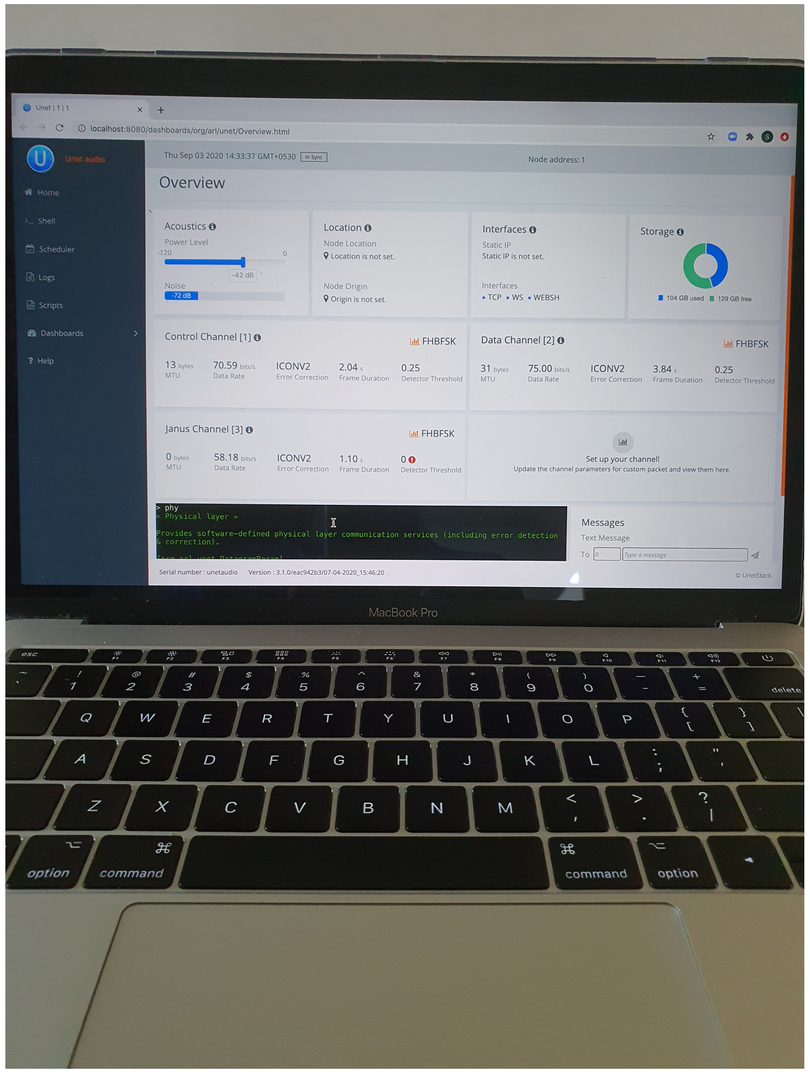
UNETSTACK

- https://unetstack.net/handbook
- Online video tutorials
  - https://www.youtube.com/watch?v=MpqhRhpwAh4
- Download and setup
  - https://unetstack.net/handbook/unet-handbook_getting_started.html



USING PC SOUND CARD

- PC sound cards offer a great way to test out basic operations
- UnetStack free community edition includes a PC audio based mode



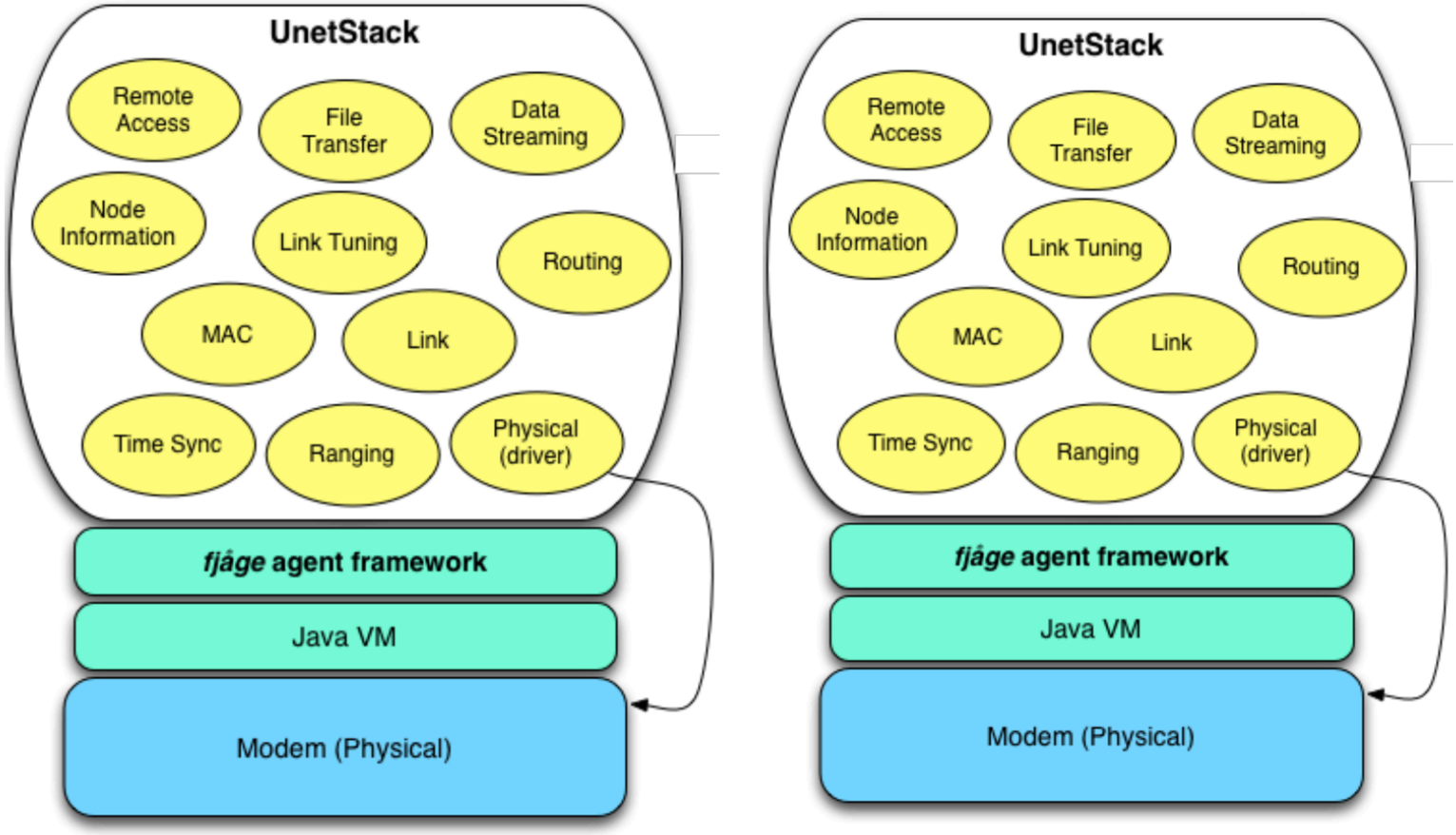
INTERFACING TO ANY MODEM

- Modem driver
  - https://blog.unetstack.net/developing-modem-drivers-for-unetstack
- Thus most concepts covered here easily ported to other modems
- Some features such as ranging and localization that requires hardware level support may not be available on all modems

AGENTS AND LAYERS

Traditional stacks are based on layers

UnetStack is based on an agent architecture



A UNIVERSAL SDM

- The variety of languages for accessing Unetstack includes C, Java, Python, Julia etc.
- Makes it easily integrated to existing software systems
- Users can replace all existing agents for different layers too
- Thus we feel overall, it is a good platform to discuss our proposed topic of Practical Underwater networks without loss of generality

PART 2

2 NODE NETWORK

- Physical layer
  - modulation - OFDM, FHBFSK
  - duplexing - time domain TDD
- Datalink
  - Reliability via acknowledgements and retransmissions
  - Propagation delay
  - Link tuning, power control

This tutorial does not aim to go into theoretical aspects such as modulation etc

2 NODE NETWORK - DEMO

```
import org.arl.fjage.*

////////////////////////////////////////////////////////////////////////////
// display documentation

println '''
2-node network
--------------

Node A: tcp://localhost:1101, http://localhost:8081/
Node B: tcp://localhost:1102, http://localhost:8082/
'''

////////////////////////////////////////////////////////////////////////////
// simulator configuration

platform = RealTimePlatform    // use real-time mode

// run the simulation forever
simulate {
  node 'A', location: [ 0.km, 0.km, -15.m], web: 8081, api: 1101, stack: "$home/etc/setup"
  node 'B', location: [ 1.km, 0.km, -15.m], web: 8082, api: 1102, stack: "$home/etc/setup"
}
```

2 NODE NETWORK - DEMO

- Connectivity - ping
- Transmissions
- Modulation parameters

```
> ping host('B')
PING 31
Response from 31: seq=0 rthops=2 time=2507 ms
Response from 31: seq=1 rthops=2 time=2852 ms
Response from 31: seq=2 rthops=2 time=2852 ms
3 packets transmitted, 3 packets received, 0% packet loss
> ping host('C')
PING 74
Response from 74: seq=0 rthops=2 time=2600 ms
Response from 74: seq=1 rthops=2 time=2634 ms
Response from 74: seq=2 rthops=2 time=2737 ms
3 packets transmitted, 3 packets received, 0% packet loss
```

A SIMPLE AGENT

- Echo Daemon

```
class EchoDaemon extends UnetAgent {

  @Override
  void startup() {
    // subscribe to all agents that provide the datagram service
    subscribeForService(Services.DATAGRAM)
  }

  @Override
  void processMessage(Message msg) {
    if (msg instanceof DatagramNtf && msg.protocol == 35) {
      // respond to protocol USER datagram
      send new DatagramReq(
        recipient: msg.sender,
        to: msg.from,
        data: msg.data
      )
    }
  }
}
```

ADDRESSING

- Node Addressing
- ports
- protocol numbers
  - Protocol.USER (32) to Protocol.MAX (63) for user applications
  - In later sections on sockets concept, this will be elaborated

3 NODE NETWORK

- Medium Access Control (MAC)

MySimplestMac.groovy

```
Message processRequest(Message msg) {
  switch (msg) {
    case ReservationReq:
      if (msg.duration <= 0)
            return new RefuseRsp(msg,'Bad reservation duration')
      ReservationStatusNtf ntf1 = new ReservationStatusNtf(
        recipient: msg.sender,
        inReplyTo: msg.msgID,
        to: msg.to,
        status: ReservationStatus.START)
      ReservationStatusNtf ntf2 = new ReservationStatusNtf(
        recipient: msg.sender,
        inReplyTo: msg.msgID,
        to: msg.to,
        status: ReservationStatus.END)
      add new OneShotBehavior({
        send ntf1
      })
      add new WakerBehavior(Math.round(1000*msg.duration), {
        send ntf2
      })
      return new ReservationRsp(msg)
    case TxAckReq:      //  by the MAC service with a RefuseRsp
      return new RefuseRsp(msg, 'Not supported')
  }
  return null
}
```

USING AUDIO

- Using PC sound

UPCOMING SESSIONS

- Multihop Routing - Prasad
- Sensors and the Internet - Chinmay
- Localization - Prasad